ETH zürich

Übung 2 - Erste Python-Programme

Informatik II

25. / 26. Februar, 2025

Heutiges Programm

Repetition der Vorlesung

Exceptions

Gruppenübung

Hausaufgaben

1. Repetition der Vorlesung

Variablen

Dynamische Typisierung[1] Variablentypen existieren in Python. Sie werden zur Laufzeit zugewiesen und nicht vom Programmierer vorab definiert.

Python

```
i = 1
```

```
d = 1.0
```

```
c = 'a'
```

```
b = True
```

C++

```
int i = 1;
```

```
double d = 1;
```

```
char c = 'a';
```

```
bool b = true;
```

[1] Dieses Thema wird zu einem späteren Zeitpunkt ausführlich besprochen.

Container

Sequences (geordnet)

- tuple
- list
- range
- string

Collections (ungeordnet)

- set
- dictionary

Operationen auf Container

Anzahl an Elementen

```
len(c)
```

Beinhaltet c x?

```
b = x in c
```

Iteration über c

```
for x in c:
    print(x)
```

Operationen auf Container

Python

Anzahl an Elementen

```
len(c)
```

Beinhaltet c x?

```
x in c
```

Iteration über c

```
for x in c:
    print(x)
```

C++

```
c.size();
```

```
std::find(c.begin(), c.end(), x);
```

```
for(int i=0;i<c.size();i++)
    std::cout << c[i] << "\n";
```

Quiz

Für alle Fragen auf dieser Folie kann folgendes angenommen werden:

c =

1	3.14	7	'a'	True
0	1	2	3	4

Was ist der Output für den folgenden Befehl?

```
len(c)
```

```
2 in c
```

```
for x in c:
    print(x)
```

Sequences

- **tuple** *(alle Objekttypen, unveränderlich)*

```
t = (0, 'a', 3.3)
```

- **list** *(alle Objekttypen, veränderlich)*

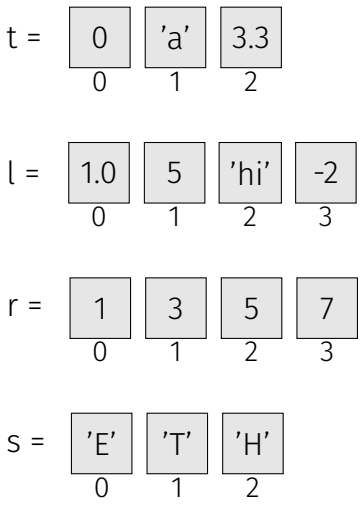
```
l = [1.0, 5, 'hi', -2]
```

- **range** *(Integers, unveränderlich)*

```
r = range(1,8,2)
```

- **string** *(Zeichen, unveränderlich)*

```
s = "ETH"
```



Operationen auf Sequences

- Subscript-Operator

```
s[i]
```

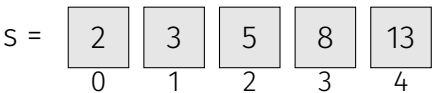
- Enumeration
  - Verbinde jedes Element mit seiner Position.

```
for (i,x) in enumerate(s):
    print(i,x)
```

- Zip
  - Verbinde zwei Sequenzen.

```
z = zip(s,t)
l = list(z)
```

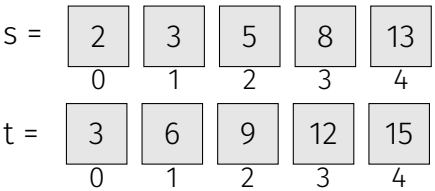
Enumeration



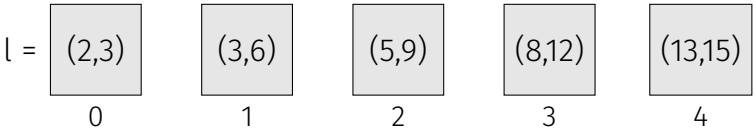

```
for (i,x) in enumerate(s):
    print(i,x)
```

```
0 2
1 3
2 5
3 8
4 13
```

Zip




```
z = zip(s,t)
l = list(z)
```



Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, Stoppe **bevor stop**, Schrittgrösse **step**

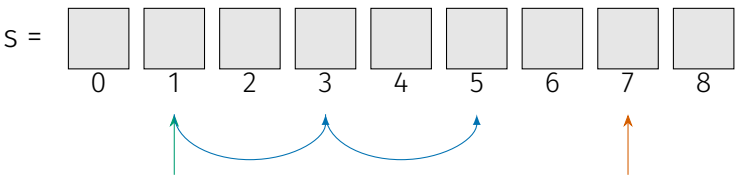
```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, stoppe **bevor stop**, Schrittgrösse **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```



s[**1**:**7**:**2**]

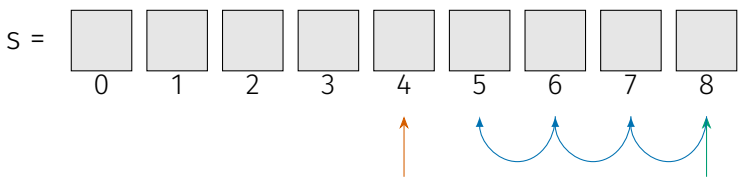
Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, stoppe **bevor stop**, Schrittgrösse **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Negativer **step**: Gehe zurück.



s[**8**:**4**:**-1**]

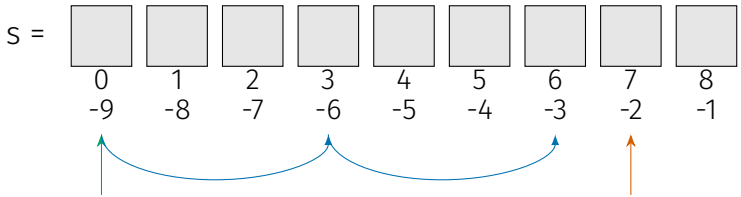
Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, stoppe **bevor stop**, Schrittgrösse **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Negative Werte **start**, stop: Verwende negative Indizes.



Slicing: Quiz

Für diese Folie kann Folgendes angenommen werden:

```
s = [1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Was ist der Output des folgenden Befehls?

```
s[3::5]
```

Wie würdest du die Sequenz s aufteilen, um den folgenden Output zu produzieren?

```
[34, 8, 2]
```

Slicing: Quiz

Es sei folgende Sequenz s gegeben:

```
s = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O']
```

Wie würdest du diese Sequenz aufteilen, um die folgenden Listen zu erzeugen?

```
['E', 'F', 'G', 'H', 'I']
['L', 'K', 'J', 'I']
['C', 'H']
['O', 'L', 'I', 'F']
```

Range

Eine Sequenz, die bei **start** beginnt, **bevor stop** endet mit der Schrittgrösse **step**.

```
range(start, stop, step)
range(start, stop) #step = 1
range(stop) #start = 0, step = 1
```

Range wird häufig in for-loops verwendet:

Python

```
for i in range(a, b, c):
    do_something
```

C++

```
for(int i=a; i<b; i+=c)
    do_something;
```

Tupel

Eine generell unveränderliche Sequenz.

```
t = () #empty tuple
t = (1) #tuple with a single element
t = (1,2) #tuple with two elements
t = tuple(range(6)) #tuple from a range
```

Quiz

Was ist der Output des folgenden Befehls?

```
tuple(range(3,15,4))
```

```
tuple(range(19,2,-2)[2:7:3])
```

Wie würdest du den folgenden Output mit einem range-Befehl generieren? Fällt dir ein anderer range-Befehl ein, der das gleiche Ergebnis erzielt? Wie viele solcher Möglichkeiten gibt es?

```
(2019, 2023, 2027)
```

Quiz

Wie würdest du den folgenden Output mit einem range-Befehl generieren?

```
[-12, -6, 0, 6, 12]
```

```
[8, 4, 0, -4]
```

Wie würdest du `range(15,-15,-3)` aufteilen, um den folgenden Output zu generieren?

```
[-9, -3, 3, 9]
```

Operationen auf Listen

- Verändern eines Elements

```
l[i] = val
```

- Anhängen eines Elements

```
l.append(val)
```

- Entfernen eines Elements

```
del l[i]
```

- Umkehren der Reihenfolge einer Liste

```
l.reverse()
```

- Liste mit k Elementen mit Wert val

```
l = [val] * k
```

Quiz

Wie sieht die Liste l aus nach jedem einzelnen Schritt?

```
l = [0] * 4
l[1] = 3
l.append(5)
l.reverse()
del l[3]
```

Collections

- **set**

```
s = {1, 6, 2, 7}
```

C++-Äquivalente: `std::set`, `std::unordered_set`

- **dictionary (dict)** *key:value*

```
d = {1:3, 6:2, 2:6, 7:5}
```

C++-Äquivalente `std::map`, `std::unordered_map`

2. Exceptions

Beispiel

- Exceptions werden ausgelöst, wenn das Programm syntaktisch korrekt ist, der Code jedoch zu einem Fehler geführt hat.
- Einige der am häufigsten vorkommenden Exceptions umfassen **IndexError**, **ImportError**, **IOError**, **ZeroDivisionError**, **TypeError**, and **FileNotFoundError**
- Das folgende Beispiel löst eine **ZeroDivisionError** Exception aus, da wir versuchen, eine Zahl durch 0 zu teilen.

```
a = 1000
b = a / 0
print(b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[1], line 2
      1 a = 1000
----> 2 b = a / 0
      3 print(b)

ZeroDivisionError: division by zero
```

Behandlung von Exceptions

- Wir können **Try**- und **Except**-Klauseln verwenden, um Exceptions zu behandeln.
- Eine try-Anweisung kann mehr als eine except-Klausel haben, um Handler für verschiedene Exceptions anzugeben.
- Es wird jedoch höchstens ein Handler ausgeführt.

```
a = [1, 2, 3]
try:
  print("Second element = %d" %(a[1]))
  # Throws error since there are only 3 elements in array
  print("Fourth element = %d" %(a[3]))
except IndexError:
  print("An error occurred")
```

Output:

```
Second element = 2
An error occurred
```

Finally

- **finally** definiert Code, der immer nach einem Try-and-Except-Block ausgeführt wird, unabhängig davon, ob eine Exception ausgelöst wird oder nicht.

```
try:
  k = 5//0 # raises divide by zero exception.
  print(k)

# handles zerodivision exception
except ZeroDivisionError:
  print("Can't divide by zero")

finally:
  # this block is always executed
  # regardless of exception generation.
  print("This is always executed")
```

Output:

```
Can't divide by zero
This is always executed
```

3. Gruppenübung

Lesen von User-Input

```
word = input("Enter a word : ")
```

- Dieser Befehl schreibt den Text "`Enter a word :`" in die Konsole und wartet auf User-Input.
- Nachdem der User Text eingegeben hat, wird dieser in der Variable `word` als String gespeichert.

Lesen von User-Input in einer Schleife

```
word = input("Enter a word : ")
  again = True
  while again:
    #Do something with word...
    word = input("Enter a word (or just <ENTER> to stop): ")
    again = len(word) > 0
```

- Dieser Code liest sequentiell String des Benutzers und verarbeitet sie.
- Wenn der Nutzer einen leeren String eingibt, wird das Programm beendet.

Gruppenübung: Palindrome

Ein **Palindrom** ist ein Wort, das vorwärts- wie rückwärtsgelesen den gleichen Sinn ergibt.

Schreibe ein Python-Programm, das:

- sequentiell Wörter (potentiell mit Leerzeichen) vom User Input einliest.
- für jedes Wort per print ausgibt, ob es sich um ein Palindrom handelt.
- Wenn der Nutzer einen leeren String eingibt, automatisch terminiert.

Wechsle zu CodeExpert - Code Examples - Exercises 2 - In-class

Hinweis: Ein String ist eine Sequence. Alle Sequence-Operationen können auf Strings angewandt werden.

Gruppenübung: Werte über dem Durchschnitt zählen

Schreibe ein Python-Programm mit dem folgenden In- und Output:[2]
Input: Eine Liste `s`, die Zahlen enthält.
Output: Die Anzahl der Zahlen in der Liste `s`, die streng grösser sind als der Durchschnittswert aller Elemente.

Example: `s = [1,1,2,3,4,1]`
Der Durchschnittswert in der Liste `s` ist gleich 2. Es gibt zwei Zahlen in `s`, die grösser sind als 2: 3 und 4. Daher sollte der Output der Funktion 2 sein.

[2] Optionale Aufgabe bei ausreichend Zeit.

4. Hausaufgaben

Aufgabe 1: Python 1

Auf https://expert.ethz.ch/mycourses/SS25/mavt2/exercises

- Sum and Maximum
- Bergprofil
- Crops & Dictionaries

Fällig bis Montag, 03.03.2025, 20:00 CET

KEIN HARDCODING

Fragen?